Separability and One-way Functions

Lance Fortnow*
John Rogers†
Department of Computer Science
The University of Chicago
1100 East 58th Street
Chicago, Illinois 60637

July 28, 1995

1 Introduction

In order to understand better the relationship between **P** and **NP**, it is useful to study the structure of languages within **NP**. In this paper we study the inseparability of **NP** and **coNP** sets and relate that to other important propositions in complexity theory.

Consider the following five propositions:

(P1) $\mathbf{P} = \mathbf{NP}$

(P2) $\mathbf{P} = \mathbf{UP}$

(P3) $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$

(P4) All disjoint pairs of **NP** sets are **P**-separable.

(P5) All disjoint pairs of **coNP** sets are **P**-separable.

At present, we do not know whether or not any of these propositions is true. In fact, for each one there is one oracle relative to which the proposition is true and another relative to which it is false so proving or refuting any of them will require nonrelativizing proof techniques.

However, we do know something about the logical relationships among them. It is relatively easy to show that (P1) implies the rest and that (P4) and (P5) each imply (P3). Grollmann and Selman [GS88] showed that (P4) also implies (P2).

Can we demonstrate any other relationships consistent with these? Our main result is that the known implications are the only ones to hold in every relativized world. That is, we demonstrate an oracle realizing each of the other consistent logical relationships.

We also look at a proposition we call Proposition Q, which states that one can easily find an accepting path of any **NP** machine that accepts Σ^*. We show a close relationship between this and (P5).

*Email: fortnow@cs.uchicago.edu. Partially supported by NSF grant CCR 92-53582.
†Currently at DePaul University. Email: rogers@cs.depaul.edu. Partially supported by NSF grant CCR 92-53582.

1.1 Generic oracles

When we have a proposition that we are unable to refute, we might attempt to construct an oracle relative to which it is true. An approach that often works is to begin by defining an infinite set of requirements so that, if each requirement is satisfied, the proposition is true. This is followed by a construction of the oracle in stages. At stage 0, we let the oracle be the empty set. At each succeeding stage, we select a requirement and add to the oracle a finite amount of information that causes the requirement to be satisfied while guaranteeing that the previously satisfied requirements stay satisfied. For example, this sort of construction can be used to build an oracle relative to which $\mathbf{P} \neq \mathbf{NP}$. For obvious reasons, such a construction is called a *finite extension* construction.

Defining the requirements is usually very easy but carrying out the construction and demonstrating that it works is often difficult. But this obscures what we are trying to accomplish. We are far more interested in whether there exists an oracle relative to which a proposition is true and far less interested in the particular form of the oracle. This is where *generic* oracles, which were first considered in complexity theory by Mehlhorn [Meh73] and later by Blum and Impagliazzo [BI87], become useful

As above, we first define a set of requirements so that if each requirement is satisfied, some proposition will be true. We then define a *generic condition*, which is usually a finite function on strings. An important contribution of this paper, following the lead of Fenner, Fortnow, Kurtz, and Li [FFKL93], is that we often can impose a special form on the conditions that makes it easier to satisfy some class of requirements. This technique played a pivotal role when Fenner, Fortnow, and Kurtz [FFK94] demonstrated an oracle relative to which the Isomorphism Conjecture holds.

We then show that, given any condition σ and any requirement, we can find another condition τ that *extends* σ and that satisfies the requirement. This demonstrates that the set of conditions that satisfy this requirement is *dense*. We then use a simple argument to show that there is an oracle that simultaneously, for every requirement, extends some condition that satisfies that requirement and so the oracle satisfies all of the requirements.

The advantage of this approach is that it focuses the proof on how to meet an individual requirement rather than on technical details about the oracle. It also usually results in a shorter and easier to understand proof.

1.2 Related Results

The history of oracle results in complexity theory begins with the paper by Baker, Gill and Solovay [BGS75] that we mentioned above. They constructed oracles A, B, C, and D such that

$$\mathbf{P}^A = \mathbf{NP}^A$$

$$\mathbf{P}^B \neq \mathbf{NP}^B = \mathbf{coNP}^B$$

$$\mathbf{P}^C \neq \mathbf{NP}^C \text{ and } \mathbf{P}^C = \mathbf{NP}^C \cap \mathbf{coNP}^C$$

$$\mathbf{P}^D \neq \mathbf{NP}^D \cap \mathbf{coNP}^D \text{ and } \mathbf{NP}^D \neq \mathbf{coNP}^D$$

Notice that, relative to A, (P1) holds so (P2) through (P5) also hold.

Rackoff [Rac82] was the first to look at relativized $\mathbf{UP}$. He created oracles A and B such that

$$\mathbf{P}^A \neq \mathbf{coNP}^A = \mathbf{NP}^A = \mathbf{UP}^A$$

$$\mathbf{P}^B = \mathbf{NP}^B \cap \mathbf{coNP}^B = \mathbf{UP}^B \neq \mathbf{NP}^B$$

Geske and Grollman [GG86] created an oracle C such that

$$\mathbf{P}^C \neq \mathbf{UP}^C \neq \mathbf{NP}^C$$

Homer and Selman [HS92] devised an oracle A such that $\mathbf{P}^A \neq \mathbf{NP}^A$ but all disjoint pairs of $\mathbf{NP}^A$ sets are $\mathbf{P}^A$-separable. We show that this is so relative to a Cohen generic oracle. They also constructed an oracle B such that $\mathbf{P}^B = \mathbf{UP}^B \neq \mathbf{NP}^B \cap \mathbf{coNP}^B$. Grollmann and Selman [GS88] showed that $\mathbf{P} = \mathbf{UP}$ iff one-way functions do not exist and that all disjoint pairs of $\mathbf{NP}$ are $\mathbf{P}$-separable iff weak one-way function do not exist. Thus, relative to B, there are weak one-way functions but no one-way functions. This result also follows as an immediate corollary of our work.

Mehlhorn was the first to consider the effect of a Cohen generic oracle G on propositions in complexity theory when he showed that $\mathbf{P}^G \neq \mathbf{NP}^G$ [Meh73]. Blum and Impagliazzo [BI87] proved, using techniques of Rackoff [Rac82] and Hartmanis and Hemachandra [HH87], that if $\mathbf{P} = \mathbf{PSPACE}$ then

$$\mathbf{P}^G = \mathbf{UP}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G \neq \mathbf{NP}^G$$

We will use some of their techniques here.

Impagliazzo and Naor [IN88] and Crescenzi and Silvestri [CS93] showed that there exists a nondeterministic Turing machine M^G accepting Σ^* for which Proposition Q fails. That is, there is no function $f^G \in \mathbf{FP}^G$ such that, for every $x \in \Sigma^*$, $f^G(x)$ is an accepting path of $M^G(x)$. This paper gives a simpler proof of that fact.

When we consider what happens relative to a random oracle R, we find that the following statements hold: $\mathbf{P}^R \neq \mathbf{NP}^R$ [BG81], $\mathbf{P}^R \neq \mathbf{UP}^R$ [KMR89, IR89], there are disjoint pairs of $\mathbf{NP}$ sets that are not $\mathbf{P}$-separable [GS88, KMR89, IR89, Ver93] and there are disjoint pairs of $\mathbf{coNP}$ sets that are not $\mathbf{P}$-separable [Ver93]. It is open whether $\mathbf{P}^R = \mathbf{NP}^R \cap \mathbf{coNP}^R$, although this is generally believed to be false.

Muchnik and Vereshchagin [MV94] have also proved various results on the relativizable relationships between separability and other propositions.

2 Definitions and Preliminaries

We define here notions that may not be widely known in complexity theory. For notions not defined here, please consult a text on formal language and complexity theory, like Hopcroft and Ullman [HU79].

2.1 Languages, oracles, and computations

Let Σ^* denote the set of all finite length binary strings and Σ^n the set of strings of length exactly n. Languages and oracles are subsets of Σ^*. Let A be an oracle and let A_n be $A \cap \Sigma^n$, that is, A_n is the set of strings of length n in A. We will sometimes abuse notation and use the name of the oracle for its characteristic function, that is, writing $A(x) = 1$ if $x \in A$ and $A(x) = 0$ otherwise.

An oracle A is *sparse* iff there is a polynomial p such that, for all n, $|A_n| \leq p(n)$. The *join* of two oracles A and B, denoted $A \oplus B$, is the oracle $\{x0 : x \in A\} \cup \{x1 : x \in B\}$.

Many of the oracles we work with here will be *gappy*. This is defined by letting *tower* be the function from integers to integers with the following recursive definition: $tower(0) = 2$, $tower(n+1) = 2^{tower(n)}$, i.e., $tower(n)$ is an exponential tower of $n+1$ 2's. An oracle A is *gappy* iff, whenever $A(x) = 1$, $|x| = tower(n)$, for some $n \in \omega$. (A notion very similar to this was used by Rackoff [Rac82].) We will call *allowed lengths* those lengths equal to $tower(n)$ for some n.

Intuitively, we make oracles gappy so that the only strings in the oracle a polynomial time computation on a particular input relies on are those at one allowed length. In polynomial time, all of the strings in the oracle at shorter allowed lengths can be enumerated and none of the strings at longer allowed lengths can be queried.

Let A be an oracle. The *zero-side* of A_n is the set of strings in A_n that end in 0. The *one-side* is the set of strings that end in 1. We say that A_n is *zero-empty* if it contains no strings ending in 0 and *one-empty* if it contains no strings ending in 1.

Fix an input x to a nondeterministic, polynomial-time oracle Turing machine M. This input generates a tree of computation paths in M. Associated with every one of these paths is a *query set*: A list of pairs of the form (q, r), where each q is an oracle query and each r a one-bit oracle response. We say that a path *expects* a string q to be in an oracle A if $(q, 1)$ is in its query set. It expects q *not* to be in the oracle if $(q, 0)$ is in its query set.

Two computation paths *conflict* iff, for some string q, one has the pair $(q, 0)$ in its query set and the other has $(q, 1)$ in its. In other words, one path expects q to be in the oracle and the other expects q not to be in the oracle. A computation path of a nondeterministic TM accepts x relative to an oracle A iff it ends in an accepting state and, for every (q, r) in its query set, $q \in A \iff r = 1$. A path is *zero-empty* iff, for every q ending in 0, $r = 0$. It is *one-empty* iff, for every q ending in 1, $r = 0$.

2.2 Language classes

In addition to the well-known language classes **P**, **NP**, **coNP**, and **PSPACE**, we will work with the function class **FP** and the language class **UP**, which we define here.

A function f is computed by a deterministic Turing machine M with a distinguished output tape if $f(x) = y$ and M, when presented with an input string x, halts with the string y on its output tape. **FP** is the class of functions computed by polynomial-time bounded deterministic Turing machines.

A function $f \in \mathbf{FP}$ is *honest* iff there is a polynomial p such that, for all x, $|x| \leq p(|f(x)|)$. That is, f does not map very long inputs to very short outputs.

A language L is in **UP** iff there is a polynomial-time bounded nondeterministic Turing machine M such that $L = L(M)$ and, for all x, if $x \in L$, the computation $M(x)$ has exactly one accepting path. Grollmann and Selman [GS88] showed that $\mathbf{P} \neq \mathbf{UP}$ iff one-way functions exist. A *one-way* function is a one-to-one, honest function in **FP** whose inverse is not in **FP**.

Let L_0 and L_1 be two languages in **NP** or in **coNP** that are disjoint, that is, $L_0 \cap L_1 = \emptyset$. We say that they are **P**-*separable* if there is a language $L \in \mathbf{P}$ such that $L_0 \subseteq L \subseteq \overline{L_1}$.

2.3 Generic conditions and oracles

Most of our oracles are *generic* oracles. These are defined in terms of some kind of generic condition, whose definition is usually designed to meet a particular set of requirements.

A *Cohen condition* is a partial function from Σ^* to $\{0, 1\}$ whose domain is finite. A condition σ *extends* another condition τ ($\sigma \supseteq \tau$) when, for all $x \in \mathrm{dom}(\tau)$, $\sigma(x) = \tau(x)$. Two conditions σ and τ are *consistent* when, for all $x \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\tau)$, $\sigma(x) = \tau(x)$. They *conflict* otherwise.

A set of Cohen conditions S is *definable* iff it can be described in a formal language powerful enough to express any of the requirements in this paper. S is *dense* iff, for every Cohen condition τ, there is a Cohen condition $\sigma \in S$ that extends τ. An oracle *meets* S if it extends at least one string in S. A *Cohen generic oracle* is one that meets at least one string in every dense, definable set of Cohen conditions.

A *size-bounded condition* is a pair (σ, k) where $k \in \omega$ and σ is a Cohen condition whose domain is gappy. Let n be an allowed length. The integer k indicates that any condition extending σ must not allow more than $n/2^k$ strings of length n to be in the oracle beyond σ. Thus, a size-bounded condition (σ, j) extends another size-bounded condition (τ, k) ($(\sigma, j) \supseteq (\tau, k)$) iff $\sigma \supseteq \tau$ and $j \geq k$ and, if $x \in \mathrm{dom}(\sigma) - \mathrm{dom}(\tau)$ then $|\{y : \sigma(y) = 1 \,\&\, |y| = |x|\}| \leq |x|/2^j$.

In the definitions of the remaining conditions, let n be a variable that ranges only over allowed lengths.

A **UP** *condition* σ is a Cohen condition whose domain is gappy and such that, for all n, $|\{y \in \Sigma^n : \sigma(y) = 1\}| \leq 1$. That is, there is at most one string in a condition at allowed lengths. A *one-sided-***UP** *condition* σ is a Cohen condition whose domain is gappy and such that, for all n, $|\{y \in \Sigma^n : \sigma(y0) = 1\}| \leq 1$. That is, it is a **UP** condition on its zero-side and a Cohen condition on its one-side.

An **NP** $\cap$ **coNP** *condition* σ is a Cohen condition whose domain is gappy. Furthermore, if σ is defined for some string z of length n then there is an x of length $n-1$ such that, for all y of length $n-1$, either:

1. $\sigma(x0) = 1$ and $\sigma(y1) = 0$, or
2. $\sigma(x1) = 1$ and $\sigma(y0) = 0$.

In other words, either the zero-side or the one-side of an allowed length (but never both) is empty.

An **NP** **P**-*inseparable condition* σ is a Cohen condition whose domain is gappy. As with **NP**$\cap$**coNP** conditions, if σ is defined for some string z of length n then either the zero-side or the one-side is empty. Unlike **NP** $\cap$ **coNP** conditions, however, both sides may be empty.

A **UP** **NP**$\cap$**coNP** *condition* σ is a Cohen condition whose domain is gappy and such that, at every length n, if there is a $z \in \Sigma^n$ such that $z \in \mathrm{dom}(\sigma)$, then

1. $|\{y : y \in \Sigma^{n-1} \,\&\, \sigma(y0) = 1\}| \leq 1$, and
2. there is an $x \in \Sigma^{n-2}$ such that, for all $y \in \Sigma^{n-2}$, either:
   (a) $\sigma(x01) = 1$ and $\sigma(y11) = 0$, or
   (b) $\sigma(x11) = 1$ and $\sigma(y01) = 0$.

Simply stated, these conditions are **UP** conditions on the zero-side and **NP** $\cap$ **coNP** conditions on the one-side.

A relativized language L^X possesses a property P *categorically* with respect to some notion of genericity if, for all oracles G of that type, L^G possesses P. For example, the language $L^X = \{0^n : (\exists x)|x| = n \,\&\, x \in X\}$ is categorically in **UP** with respect to a **UP**-generic G because G contains at most one string at every length.

2.4 Propositions Q and Q′

Now we introduce a notion closely related to **P**-separability in **NP** and **coNP** which we call Proposition Q. It has many characterizations and we present several here.

Proposition Q.

Q1. For all **NP** machines M such that $L(M) = \Sigma^*$, there is a function $f \in$ **FP** such that, for all $x \in \Sigma^*$, $f(x)$ is an accepting path of M.

Q2. For all **NP** machines M such that $L(M) \in \mathbf{P}$, there is a function $f \in \mathbf{FP}$ such that, for all $x \in L(M)$, $f(x)$ is an accepting path of M.

Q3. For all sets S of CNF propositional formulae such that $S \subseteq \mathbf{SAT}$ and $S \in \mathbf{P}$, there is a function $f \in \mathbf{FP}$ such that, for all $\varphi \in S$, $f(\varphi)$ is a satisfying assignment of φ.

Q4. For every honest onto function $f \in \mathbf{FP}$, there is a function $g \in \mathbf{FP}$ such that, for all x, $f \circ g(x) = x$.

We will also work with a closely related notion called Proposition Q′. It also has a number of characterizations, including one involving separability, and we show that they are equivalent in the following lemma.

Lemma 2.1 *The following are equivalent:*

1. *All disjoint* **coNP** *sets are* **P***-separable.*
2. *For every onto, honest* $f \in \mathbf{FP}$*, there is a* $g \in \mathbf{FP}$ *whose range is* Σ *and such that, for all* x*,* $g(x)$ *is the first bit of a* y *such that* $f(y) = x$*.*
3. *For every onto, honest* $f \in \mathbf{FP}$ *and every constant* k*, and every* $p : \Sigma^* \to \Sigma^k \in \mathbf{FP}$*, there is a* $g : \Sigma^* \to \Sigma^k \in \mathbf{FP}$ *such that, for all* $x \in \Sigma^*$*,* $g(x) = p(y)$*, for a* y *such that* $f(y) = x$*.*
4. *For every two NP sets* L_1 *and* L_2 *such that* $L_1 \cup L_2 = \Sigma^*$*, there is a function* $g : \Sigma^* \to \{1, 2\} \in \mathbf{FP}$ *such that* $g(x) = i$ *implies that* $x \in L_i$ *for all* $x \in \Sigma^*$*.*
5. *For every* k *NP sets* $L_1, \ldots, L_k$ *such that* $\cup_{1 \leq i \leq k} L_i = \Sigma^*$*, there is a function* $g : \Sigma^* \to \{1, \ldots, k\} \in$ **FP** *such that* $g(x) = i$ *implies that* $x \in L_i$ *for all* $x \in \Sigma^*$*.*

Proof. (1⇒4). Let L_1 and L_2 be two **NP** sets such that $L_1 \cup L_2 = \Sigma^*$. Then $\overline{L_1}$ and $\overline{L_2}$ are two disjoint **coNP** sets. Therefore there is a language $L \in \mathbf{P}$ such that $\overline{L_1} \subseteq L \subseteq L_2$. Let $g(x) = 2$ if $x \in L$ and $g(x) = 1$ if $x \notin L$.

(4⇒5). By induction on k. The base case $k = 2$ is the assumption. Assume true for $j \leq k$. Let $L_1, \ldots, L_{k+1}$ be a collection of NP sets such that $\cup_{1 \leq i \leq k+1} L_i = \Sigma^*$. Let M_i be the **NP** machine that accepts L_i for each i, $1 \leq i \leq k+1$.

Let $M'(x)$ be the **NP** machine that accepts x if $M_i(x)$ accepts for $1 \leq i \leq k$. $L(M'(x))$ and L_{k+1} are two **NP** languages whose union is Σ^*. Let g_1 be the function that exists by the induction hypothesis.

Let $M'_{k+1}(x)$ be the machine that simulates $M_{k+1}(x)$ except that it rejects immediately if $g_1(x) = 1$. Let $M'_k(x)$ accept if either $M_k(x)$ accepts or $M'_{k+1}(x)$ accepts. $L_1, \ldots, L_{k-1}, L(M'_k)$ are a collection of k **NP** machines whose union is Σ^*. Let g_2 be the function that arises from the induction hypothesis.

Let $g(x) = k + 1$ if $g_1(x) = 2$. Otherwise let $g(x) = g_2(x)$.

(5⇒3). For each $w \in \Sigma^k$, let

$$L_w = \{x \mid \text{For some } y \in \Sigma^*,\ p(y) = w \text{ and } f(y) = x\}$$

We then have that L_w is a collection of 2^k **NP** sets whose union is Σ^*. The g that arises from the assumption will work virtually unchanged for the conclusion.

(3⇒2). Clear.

(2⇒1). Let L_1 and L_2 be two disjoint **coNP** sets. Let M_1 and M_2 be the **NP** machines that accept $\overline{L_1}$ and $\overline{L_2}$ respectively. Let $f(0\langle x, p\rangle 0) = 0x$ if p is an accepting path of $M_1(x)$ and $f(0\langle x, p\rangle 0) = 1\langle x, p\rangle$ otherwise. Likewise let $f(1\langle x, p\rangle 0) = 0x$ if p is an accepting path of $M_2(x)$ and $f(1\langle x, p\rangle 0) = 1\langle x, p\rangle$ otherwise. Finally let $f(z1) = 1z$ for all $z \in \Sigma^*$. Clearly f is honest and onto.

Let g be the function that arises from the assumption using f. Define L as the set of x such that $g(0x) = 1$. One can now easily check that $L_1 \subseteq L \subseteq \overline{L_2}$ ■

Finally, we tie Propositions Q and Q$'$ together.

Corollary 2.2 *Proposition Q implies Proposition Q$'$ (and so implies that all disjoint* **coNP** *sets are* **P***-separable).*

It is open whether Q$'$ implies Q. Please see [FFNR95] for a more thorough treatment of Propositions Q and Q$'$.

Because Q implies Q$'$, we will use the following lemma, with corollary 2.2, to show that all disjoint **coNP** sets are **P**-separable, relative to certain oracles.

Lemma 2.3 *Assume* $\mathbf{P} = \mathbf{PSPACE}$ *and let S be a sparse oracle. If M^S is an* $\mathbf{NP}^S$ *machine such that, for all $T \subseteq S$, $L(M^T) = \Sigma^*$ then, relative to S, Proposition Q holds, i.e., there is a function* $f^S \in \mathbf{FP}^S$ *such that, for all x, $f^S(x)$ is an accepting path of $M^S(x)$.*

Proof. Let $f(x)$ be the function defined by the algorithm in figure 1. The algorithm uses the $\mathbf{P} = \mathbf{PSPACE}$ assumption to find accepting paths of $M^\alpha(x)$ and uses S to determine when it has found strings in S.

Now we need to show that the algorithm runs correctly in polynomial time. By our assumption that, for all $T \subset S$, $L(M^T) = \Sigma^*$, it is easy to see that $L(M^\alpha)$ is always Σ^* in the algorithm. If an accepting path p found by the algorithm is not an accepting path in $M^S(x)$, then there is some z in S such that $(z, 0)$ is in p's query set. Because S is sparse, this can happen at most a polynomial number of times before the algorithm determines every string in the oracle at the lengths that can be queried for input x. But then, the while-loop terminates because the algorithm will find an accepting path of $M^S(x)$.

So $f \in \mathbf{FP}^S$ and, given an input x, it finds an accepting path of M^S on x. ■

```
Let α = ∅.
While there is an accepting path p of M^α(x) do
    If p is an accepting path of M^S(x) then
        Output p and halt
    Else
        Let z ∈ S − α be a string queried by M^S(x) on path p;
        Let α := α ∪ {z};
    Endif
Endwhile
```

Figure 1: Algorithm used in lemma 2.3 to compute $f^S(x)$

Every sparse oracle S in this paper fulfills the requirements of lemma 2.3.

2.5 A pair of disjoint languages in coNP

We will now define a pair of categorically disjoint languages in **coNP** that, relative to certain oracles we use below, will be **P**-inseparable.

We begin by defining the following two relativized functions:

$$f^X(x) = X(x0^{|x|-1}1)X(x0^{|x|-2}11)\ldots X(x1^{|x|})$$

$$h^X(x,y,z) = \begin{cases} 0^{|x|+|y|+|z|} & \text{if } x,y,z \text{ are not all the same length} \\ f^X(x) & \text{if } f(y)=f(z) \Rightarrow y=z \\ x & \text{if } y \neq z \text{ and } f(y)=f(z) \end{cases}$$

For all X, f^X is length preserving and all the strings it depends on in X are the same length and end in 1. Also, for all X, h^X is honest. We also have:

Lemma 2.4 *For all oracles X, $rng\, h^X = \Sigma^*$.*

Proof. It is enough to show that, for all n, $\Sigma^n \subset \operatorname{rng} h^X$. Fix an n.

Case 1. On length n, f^X is a permutation. So, for every $w \in \Sigma^n$, there is an $x \in \Sigma^n$ such that $f^X(x) = w$. Thus $h^X(x, 0^{|x|}, 0^{|x|}) = w$.

Case 2. On length n, f^X is not a permutation. Then there are $y_0, z_0 \in \Sigma^n$ such that $y_0 \neq z_0$ and $f^X(y_0) = f^X(z_0)$. So, for every w, $h^X(w, y_0, z_0) = w$. ■

We next define the following two languages:

$$\begin{aligned} L_0^X &= \{w : (\exists x,y,z)h^X(x,y,z) = w \text{ and } x \text{ begins with } 0\} \\ L_1^X &= \{w : (\exists x,y,z)h^X(x,y,z) = w \text{ and } x \text{ begins with } 1\} \end{aligned}$$

Because h is honest, we can bound the lengths of x, y, and z by a polynomial in the length of w so these languages are in $\mathbf{NP}^X$. Their union contains $\operatorname{rng} h^X$ and so, by lemma 2.4, their union is categorically Σ^*. Let $C_0^X = \overline{L_0^X}$ and $C_1^X = \overline{L_1^X}$. The C_i form a pair of categorically disjoint $\mathbf{coNP}^X$ languages.

To make it easier to refer to these languages, here are their explicit definitions:

$$\begin{aligned} C_0^X &= \{w : (\forall x,y,z)h^X(x,y,z) = w \Rightarrow x \text{ begins with } 1\} \\ C_1^X &= \{w : (\forall x,y,z)h^X(x,y,z) = w \Rightarrow x \text{ begins with } 0\} \end{aligned}$$

The following lemma summarizes the known relationships among Propositions (P2) through (P5).

Lemma 2.5 *1. If there are no* **P***-inseparable sets in* **coNP** *then* $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$.

2. If there are no **P***-inseparable sets in* **NP** *then*

   (a) $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$;

   (b) $\mathbf{P} = \mathbf{UP}$.

Proof. Items 1 and 2a are easy to see. The proof of item 2b is found in [GS88]. ■

We end this section by mentioning that, relative to all of the oracles we construct, $\mathbf{P} \neq \mathbf{NP}$. This is true because each of these oracles allows us to use the same diagonalization used by Baker, Gill, and Solovay [BGS75] to create a language in **NP** but not in **P**.

3 Results

Throughout this section, we assume that all statements are made relative to an oracle H such that $\mathbf{P}^H = \mathbf{PSPACE}^H$. For example, we can let H be the characteristic function for the **PSPACE**-complete language Quantified Boolean Formulae. We can make this assumption because all of the proofs in this section relativize, that is, if there is an oracle relative to which a theorem's hypotheses are true then the proof given here is sufficient to establish the truth of the conclusion relative to that oracle. So a theorem stated as:

Let G be an X-generic oracle. Then the following propositions are true:. . . .

$\mathbf{P} = \mathbf{UP}$	$\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$	No **NP** **P**-insep	No **coNP** **P**-insep	Generic oracle	Thm
Y	Y	Y	Y	size-bounded (sb)	3.1
Y	Y	Y	N	Cohen	3.2
Y	Y	N	Y	sb-**NP** **P**-insep	3.4
Y	Y	N	N	**NP** **P**-insep	3.3
Y	N	Y	Y	inconsistent	–
Y	N	Y	N	inconsistent	–
Y	N	N	Y	inconsistent	–
Y	N	N	N	$\mathbf{NP} \cap \mathbf{coNP}$	3.5
N	Y	Y	Y	inconsistent	–
N	Y	Y	N	inconsistent	–
N	Y	N	Y	**UP**	3.6
N	Y	N	N	one-sided-**UP**	3.7
N	N	Y	Y	inconsistent	–
N	N	Y	N	inconsistent	–
N	N	N	Y	inconsistent	–
N	N	N	N	**UP** $\mathbf{NP} \cap \mathbf{coNP}$	3.8

Table 1: Summary of results

should be interpreted as:

> *Let H be an oracle relative to which* $\mathbf{P} = \mathbf{PSPACE}$*. Let G be an oracle that is X-generic with respect to H. Then the following propositions are true:. . . .*

Among the four propositions (P2) through (P5), there are sixteen possible combinations for which ones are true and which are false. We call a combination *consistent* if it is logically consistent with the implications we presently know. This leaves eight consistent combinations. Our contribution is to show that, for each of these eight, there is an oracle relative to which it holds. This is important because it demonstrates that any relationship not already known among propositions (P2) to (P5) will require techniques that do not relativize. This is summarized in table 1 which lists, for every consistent combination, a type of generic oracle and a theorem number. Let G denote one of these generic oracles. The theorem indicated proves that the corresponding combination holds relative to an oracle $H \oplus G$, where H is as above.

To separate a class from $\mathbf{P}$, we usually define a language L that depends on the generic oracle and that gives us, for any polynomial time deterministic oracle Turing machine M, an infinite number of chances to prevent M from accepting L. That is, we have an infinite number of opportunities to diagonalize against M.

To make a class equal to $\mathbf{P}$, we demonstrate for each language L in the class a Turing machine algorithm that accepts L and runs in deterministic polynomial time relative to a particular kind of generic oracle. This algorithm will, to some degree, be a variation on what we call the "usual algorithm." Let $C = H \oplus G$, where G is Cohen generic. The "usual algorithm" refers either to the algorithm in [BI87] used to show that $\mathbf{P}^C = \mathbf{NP}^C \cap \mathbf{coNP}^C$ and that all pairs of disjoint $\mathbf{NP}^C$ sets are $\mathbf{P}^C$-separable or to the algorithm in [HH87, BI87] used to show that $\mathbf{P}^C = \mathbf{UP}^C$.

The first theorem demonstrates what happens to the propositions relative to a Cohen generic oracle. The remaining theorems demonstrate what happens relative to oracles that are generic with respect to a wide variety of conditions. In the proofs of each of those theorems, we will define the condition that gives rise to the oracle needed.

Theorem 3.1 *Let G be a size-bounded generic oracle. Then*

1. $\mathbf{P}^G = \mathbf{UP}^G$;
2. $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are no* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{NP}^G$;
4. *there are no* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{coNP}^G$.

Proof. A size-bounded generic oracle fulfills the hypotheses of lemma 2.3 so Proposition Q holds with respect to G. By this and corollary 2.2, all disjoint pairs of $\mathbf{coNP}^G$ sets are $\mathbf{P}^G$-separable.

To see that there are no $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$, let M_0 and M_1 be a pair of categorical $\mathbf{NP}$ machines whose languages, with respect to size-bounded conditions, are categorically disjoint. Let (τ, k) be a size-bounded condition. Extend it with the condition (σ, j) where $\sigma = \tau$ and $j = k + 1$. For all but a finite number of x, an accepting path in either machine must require that there be no more than $|x|/2^j$ strings in the oracle. So we can use the usual algorithm but ignore any accepting path requiring G to have more than $|x|/2^j$ strings in it. The algorithm works because, if there is a σ such that there are accepting paths in both $M_0^\sigma(x)$ and $M_1^\sigma(x)$, these paths must conflict. If they were consistent with one another, (τ, k) could have been extended by a condition (τ', k) that would force both machines to accept x. But this violates the assumption of categorical disjointness.

Because there are no $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$, $\mathbf{P}^G = \mathbf{UP}^G$ and $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$ by lemma 2.5. ■

Theorem 3.2 *Let G be a Cohen generic oracle. Then*

1. $\mathbf{P}^G = \mathbf{UP}^G$;
2. $\mathbf{P} = \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are no* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{NP}^G$;
4. *there are* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{coNP}^G$.

Proof. The first two propositions are shown in [BI87] and [HH87]. The third follows by using essentially the same technique as that used to show that $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$.

To see that there are $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$, let C_0^X and C_1^X be the languages and f^X the function defined in section 2.5. We now show that C_0^G and C_1^G are not $\mathbf{P}^G$-separable. Let τ be a Cohen condition and let M be a deterministic polynomial-time oracle machine running in time $p(|x|)$. Fix an even length n at which τ is not yet defined and $p(n) < 2^n$. Select two strings w_0 and w_1 of length $n/2$. Run M^τ on w_0 and then on w_1, answering queries outside of τ's domain in such a way as to allow $f^{\tau'}$ to realize a partial permutation on the strings of length $n/2$ that maps no string to either w_0 or w_1. Call the resulting extension τ'. Such an extension exists because M^τ can only query a polynomial number of strings.

If $M^{\tau'}$ accepts or rejects both strings, extend τ' to a σ that makes f^σ a permutation mapping a string starting with 0 to w_0, which puts w_0 into C_0^σ, and a string starting with 1 to w_1, which puts w_1 into C_1^σ. If $M^{\tau'}$ accepts one and rejects the other, extend τ' to a σ that makes f^σ a permutation mapping two different strings, both starting with 0, to w_0 and w_1, thus forcing $w_0, w_1 \in C_0^\sigma$. In either case, we force $L(M^G)$ not to be a $\mathbf{P}^G$-separator of C_0^G and C_1^G. ■

Theorem 3.3 *Let G be an* **NP P***-inseparable generic oracle. Then*

1. $\mathbf{P}^G = \mathbf{UP}^G$;

2. $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$;

3. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$;*

4. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$.*

Proof. We first show that $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$. Let M_0 and M_1 be $\mathbf{NP}^G$ machines that accept languages that are categorically complementary with respect to **NP P**-inseparable conditions. Fix an input x and let n be the only relevant allowed length. We will demonstrate an algorithm that accepts x iff $x \in L(M_0^G)$.

If we knew whether there were any strings of length n in G and, if so, whether they all ended in 0 or 1, we could run the usual algorithm under this assumption and get the correct result.

Because *a priori* we do not know, the algorithm must make some assumptions. It assumes first that the allowed length is empty and so checks whether there is an accepting path P in $M_0^G(x)$ that expects no strings in the oracle at the allowed length. There are two cases:

Case 1. There is such a P. Make all queries along P to G. If these queries do not find a string at the allowed length then P is a valid accepting path relative to G so accept. If some query finds a string ending in 0 then run the usual algorithm under the correct assumption that every string in the oracle at the allowed length ends in 0. Do likewise if a string ending in 1 is found.

Case 2. There is no such P. Then there must be a path P' in $M_1^G(x)$ that expects no strings in the oracle at the allowed length. If there were not, the machines could have both been forced to reject x by the oracle having no strings at the allowed length, which violates their categorical complementariness. Make all queries along P' to G. If P' is indeed an accepting path then reject. If it is not an accepting path then, as above, the algorithm finds a string in G at the allowed length and uses that to run the usual algorithm under the correct assumption.

Next we show that $\mathbf{P}^G = \mathbf{UP}^G$. Let M be a machine that is categorically **UP** with respect to **NP P**-inseparable conditions. Fix an input x. Run the usual algorithm under the assumption that the allowed length is one-empty, that is, only consider one-empty accepting paths in $M^G(x)$. If an accepting path is found, accept. If not, instead of outright rejecting, run the usual algorithm again but this time under the assumption that the allowed length is zero-empty so only consider zero-empty accepting paths. If an accepting path is found, accept, otherwise reject. This works because the allowed length is either zero-empty or one-empty so if $M^G(x)$ has an accepting path, it will be either zero-empty or one-empty.

Now we show that there is a pair of $\mathbf{P}^G$-inseparable languages in $\mathbf{NP}^G$. Let $L_i^X = \{0^n : (\exists y)[|y| = n-1 \;\&\; yi \in X]\}$, for $i \in \{0,1\}$. Relative to G, L_0^G and L_1^G are a pair of disjoint $\mathbf{NP}^G$ sets. Let τ be an **NP P**-inseparable condition and let M be a deterministic polynomial time oracle machine running in time $p(|x|)$. Let n be an allowed length on which τ is undefined and such that $p(n) < 2^{n-1}$. Let m be an allowed length greater than n. Run M once on input 0^n and once on input 0^m. Extend the oracle so that those strings of length n or m queried by M are left out.

Because M cannot query every string of length n or m ending in 0 or every string ending in 1, we can do the following: If M accepts both or rejects both inputs, set unqueried strings at lengths n and m so as to put 0^n into L_0 and 0^m into L_1. If M accepts one and rejects the other, set the unqueried strings so as to put both 0^n and 0^m into L_0. This forces $L(M)$ not to be a $\mathbf{P}^G$-separator of L_0^G and L_1^G.

Finally, we show that there are $\mathbf{P}^G$-inseparable languages in $\mathbf{coNP}^G$. Let C_0^X and C_1^X be the languages and f^X the function defined in section 2.5. The definition of f^X allows us, using the same argument as in theorem 3.2, to extend one **NP P**-inseparable condition by another that forces a particular polynomial-time deterministic oracle machine not to be a $\mathbf{P}^G$-separator of C_0^G and C_1^G. ∎

Theorem 3.4 *Let G be a size-bounded* **NP P***-inseparable generic oracle. Then*

1. $\mathbf{P}^G = \mathbf{UP}^G$;
2. $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$;*
4. *there are no $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$.*

Proof. First we show that there are no $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$. Because a size-bounded generic fits the hypotheses of lemma 2.3, Proposition Q holds with respect to G so, by corollary 2.2, all disjoint pairs of $\mathbf{coNP}^G$ sets are $\mathbf{P}^G$-separable. By this and lemma 2.5, $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$.

Next we show that there is a pair of $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$. Let L_0^X and L_1^X be the languages defined in theorem 3.3. We can use the argument there to show that L_0^G and L_1^G are in $\mathbf{NP}^G$ and $\mathbf{P}^G$-inseparable.

Finally we show that $\mathbf{P}^G = \mathbf{UP}^G$. Let M be an **NP** machine that is categorically **UP** with respect to size-bounded **NP P**-inseparable conditions. Let (τ, k) be a size-bounded condition. Extend it with the condition (σ, j) where $\sigma = \tau$ and $j = k + 1$. For all but a finite number of x, an accepting path in $M^G(x)$ must require that there be no more than $|x|/2^j$ strings in the oracle. So we can use the usual algorithm but ignore any accepting path requiring G to have more than $|x|/2^j$ strings in it. The algorithm works because, if there is a σ such that there are two such accepting paths in $M^\sigma(x)$, these paths must conflict. If they were consistent with one another, (τ, k) could have been extended by a condition (τ', k) that would force $M(x)$ to have two accepting paths, violating the assumption that it is categorically **UP**. ∎

Theorem 3.5 *Let G be an* $\mathbf{NP} \cap \mathbf{coNP}$*-generic oracle. Then*

1. $\mathbf{P}^G = \mathbf{UP}^G$;
2. $\mathbf{P}^G \neq \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$*
4. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$.*

Homer and Selman [HS92] created an oracle that made the consequences of theorem 3.5 true. We will give a simple proof using generics.

Proof of theorem 3.5. Because an allowed length in G is either zero-empty or one-empty, we can show that $\mathbf{P}^G = \mathbf{UP}^G$ in exactly the same way we showed it in theorem 3.3.

Next we show that $\mathbf{P}^G \neq \mathbf{NP}^G \cap \mathbf{coNP}^G$. Let $L_0^X = \{0^n : (\exists y)[|y| = n - 1 \;\&\; y0 \in X]\}$ and $L_1^X = \{0^n : (\forall y)[|y| = n - 1 \Rightarrow y1 \notin X]\}$. L_0^X is categorically in $\mathbf{NP}^X$ and L_1^X is categorically in $\mathbf{coNP}^X$. By the definition of G, $L_0^G = L_1^G$ so $L_0^G \in \mathbf{NP}^G \cap \mathbf{coNP}^G$. let τ be an $\mathbf{NP} \cap \mathbf{coNP}$ condition τ and M a deterministic polynomial-time oracle machine running in time $p(|x|)$. Let n be an allowed length on which τ is not defined and such that $p(n) < 2^{n-1}$. Run M on input 0^n. There is a string x of length n ending in 0 not queried by $M(0^n)$ and a string y of length n ending in 1 not queried.

If M accepts 0^n, let σ be $\tau \cup \{y\}$. If M rejects 0^n, let σ be $\tau \cup \{x\}$. In both cases, let x or y be the only string in σ of length n. We now have that $M^\sigma(0^n)$ accepts iff $0^n \notin L_0^\sigma$.

That there are $\mathbf{P}^G$-inseparable pairs of sets in both $\mathbf{NP}^G$ and $\mathbf{coNP}^G$ follows from lemma 2.5. ■

Theorem 3.6 *Let G be a* **UP***-generic oracle. Then*

1. $\mathbf{P}^G \neq \mathbf{UP}^G$;
2. $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{NP}$;
4. *there are no* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{coNP}^G$.

Proof. First we show that $\mathbf{P}^G \neq \mathbf{UP}^G$. Let $L^X = \{0^n : (\exists y)[|y| = n \,\&\, y \in X]\}$. By the definition of G, $L^G \in \mathbf{UP}^G$. Let τ be a **UP**-condition and let M be a deterministic polynomial-time oracle machine running in time $p(|x|)$. Let n be a length on which τ is not defined and such that $p(n) < 2^n$. Run M on input 0^n, responding 0 to any queries of strings of length n. There will be some x of length n that is not queried. If M accepts, let σ be τ with every string at length n left out. If M rejects, let σ be $\tau \cup \{x\}$ with every other string of length n left out. We then have that $M^\sigma(0^n)$ accepts iff $0^n \notin L^\sigma$.

By lemma 2.5, there is a pair of $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$.

The hypotheses of lemma 2.3 hold for a **UP**-generic so proposition Q holds with respect to G. By corollary 2.2, all disjoint pairs of $\mathbf{coNP}^G$ sets are $\mathbf{P}^G$-separable. By lemma 2.5, $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$.
■

Theorem 3.7 *Let G be a one-sided* **UP***-generic oracle. Then*

1. $\mathbf{P}^G \neq \mathbf{UP}^G$;
2. $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{NP}^G$
4. *there are* $\mathbf{P}^G$*-inseparable sets in* $\mathbf{coNP}^G$.

Proof. First we show that $\mathbf{P}^G = \mathbf{NP}^G \cap \mathbf{coNP}^G$. Let M_0 and M_1 be $\mathbf{NP}^G$-machines whose languages, with respect to one-sided **UP**-generic conditions, are categorically complementary. Fix an input x. We will demonstrate that the following polynomial time algorithm accepts x iff $x \in L(M_0^G)$.

As before, G is gappy so at most one allowed length can affect the computations $M_0^G(x)$ and $M_1^G(x)$. The algorithm may ignore any accepting path expecting more than one string in the oracle on the zero-side. If there is a string on the zero-side of the allowed length and the algorithm finds it, we can run the usual algorithm using this information and get a correct answer.

Let γ be the set of the strings whose membership or nonmembership in G has been determined. Initialize γ to $\emptyset$.

Step 1. In this step, we try to determine which machine has a zero-empty accepting path P that is compatible with the one-side of the allowed length. Now P must exist. If it did not, every accepting path in both machines expects at least one string in the oracle on the zero-side of the allowed length. Then G could force both machines to reject x by leaving that zero-side empty. But this contradicts the assumption that the languages are categorically complementary.

Run the usual algorithm under the assumption that the zero-side of the allowed length is empty, that is, only consider zero-empty accepting paths in $M_0^\gamma(x)$. If there is a zero-empty accepting path in

$M_0^G(x)$, the algorithm will find it in polynomial time. If we run out of zero-empty accepting paths in $M_0^\gamma(x)$ then P must be in $M_1^G(x)$. Go to step 2.

If we do not run out of accepting paths in $M_0^\gamma(x)$, upon exiting the loop, we check whether there is an accepting path P' in $M_1^\gamma(x)$ that, on the one-side, is compatible with γ and requires no other queries.

If so, query the oracle on those strings P' queries on the zero-side. Then either we find a string there, so we run the usual algorithm with this assumption, or we do not find a string, in which case this is an accepting path in $M_1^G(x)$ so reject.

If P' does not exist then P is not in $M_1^G(x)$. If it were, we would have found it because every zero-empty path in $M_0^\gamma(x)$ must conflict with P and we would have queried enough of the oracle to find everything P queries on the one-side. So P must be in $M_0^G(x)$. Go to step 2.

Step 2. We now know which computation has P. Without loss of generality, assume that it is $M_0^G(x)$. Let ${M'_0}^G(x)$ be $M_0^G(x)$ with any accepting paths expecting a string on the zero-side removed. Note that ${M'_0}^G(x)$ still has P.

Run the usual algorithm on ${M'_0}^\gamma(x)$ and $M_1^\gamma(x)$ with no assumptions, extending the γ from step 1. Each accepting path in $M_1^\gamma(x)$ must conflict with P in ${M'_0}^\gamma(x)$ so, as usual, γ acquires the strings queried along P (or a similar path). If this were not true, there would be a zero-empty accepting path P in ${M'_0}^G(x)$ (and hence $M_0^G(x)$) and some accepting path P_1 in $M_1^G(x)$ that expects at most one string on the zero-side of the allowed length such that P and P_1 are compatible. But then their union forms a one-sided **UP**-condition so G could have forced both to accept x, contradicting their categorical disjointness.

If we run out of accepting paths in $M_1^\gamma(x)$ then accept. Otherwise, check whether there is a zero-empty accepting path in ${M'_0}^\gamma(x)$. If so, accept because ${M'_0}^G(x)$, and so $M_0^G(x)$, accepts. If not, then P, which has all of its queries on the one-side in γ, must conflict with the oracle on the zero-side. But then, because it is zero-empty, the conflict must be because one of the strings it queries on the zero-side is in the oracle. We can then use this to run the usual algorithm.

Now we show that $\mathbf{P}^G \neq \mathbf{UP}^G$. Let $L^X = \{0^n : (\exists y)[|y| = n-1 \ \&\ y0 \in X]\}$. Using virtually the same argument as in theorem 3.6, we can show that $L^G \in \mathbf{UP}^G - \mathbf{P}^G$. By lemma 2.5, there are no $\mathbf{P}^G$-separable sets in $\mathbf{NP}^G$.

Finally, we show that there are $\mathbf{P}^G$-inseparable languages in $\mathbf{coNP}^G$. Let C_0^X and C_1^X be the languages and f^X the function defined in section 2.5. We can use the same argument as in theorem 3.2 to show that we can always extend one one-sided **UP** condition by another that forces a particular polynomial time deterministic machine M not to be a $\mathbf{P}^G$-separator of C_0^G and C_1^G. ■

Theorem 3.8 *Let G be a* **UP NP** $\cap$ **coNP**-*generic oracle. Then*

1. $\mathbf{P}^G \neq \mathbf{UP}^G$;
2. $\mathbf{P}^G \neq \mathbf{NP}^G \cap \mathbf{coNP}^G$;
3. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{NP}^G$;*
4. *there are $\mathbf{P}^G$-inseparable sets in $\mathbf{coNP}^G$.*

Proof. First we show that $\mathbf{P}^G \neq \mathbf{UP}^G$. Let $L^X = \{0^n : (\exists y)[|y| = n-1 \ \&\ y0 \in X]\}$. By the same argument as in theorem 3.7, $L^G \in \mathbf{UP}^G - \mathbf{P}^G$.

Next we show that $\mathbf{P}^G \neq \mathbf{NP}^G \cap \mathbf{coNP}^G$. Let $L_0^X = \{0^n : (\exists y)[|y| = n-2 \ \& \ y01 \in X]\}$ and $L_1^X = \{0^n : (\forall y)[|y| = n-2 \Rightarrow y11 \in X]\}$. Using essentially the same argument as in theorem 3.5, $L_0^G \in (\mathbf{NP}^G \cap \mathbf{coNP}^G) - \mathbf{P}^G$.

By lemma 2.5, there are no $\mathbf{P}^G$-separable sets in either $\mathbf{NP}^G$ or $\mathbf{coNP}^G$. ∎

4 Conclusions and open problems

Two very interesting open questions remain from this paper. The first is whether all disjoint pairs of **coNP** sets being **P**-separable (Proposition Q′) implies that we can find accepting paths for **NP** machines that accept Σ^* (Proposition Q). Every oracle that we construct in this paper makes these propositions either both true or both false. More information about both propositions may be found in Fenner, Fortnow, Naik, and Rogers [FFNR95].

Also interesting is the question of whether $\mathbf{P} = \mathbf{UP}$ and $\mathbf{NP} = \mathbf{coNP}$ implies $\mathbf{P} = \mathbf{NP}$. Perhaps some form of genericity that preserves $\mathbf{P} = \mathbf{UP}$ and $\mathbf{NP} = \mathbf{coNP}$ but separates **P** from **NP** could be found. Homer and Selman [HS92] have the best known results along these lines showing that there exists oracles A and B such that

$$\mathbf{P}^A = \mathbf{UP}^A \neq \Pi_2^A = \Sigma_2^A$$

and

$$\mathbf{UP}^B \neq \mathbf{NP}^B = \mathbf{coNP}^B.$$

We should also mention that results similar to ours were proven independently by Muchnik and Vereshchagin [MV94].

Acknowledgments

We would like to thank Stuart Kurtz, Sanjeev Saluja and Richard Beigel for helpful discussion on these questions and Sophie Laplante for helping to make the exposition clearer.

Some of this material appeared as part of [Rog95].

References

[BG81] C. Bennet and J. Gill. Relative to a random oracle, $P^A \neq NP^A \neq co{-}NP^A$ with probability one. *SIAM Journal on Computing*, 10:96–113, 1981.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P = NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 118–126. IEEE, New York, 1987.

[CS93] P. Crescenzi and R. Silvestri. Sperner's lemma and robust machines. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 194–199. IEEE, New York, 1993.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 120–131. IEEE, New York, 1993.

[FFNR95] Steven Fenner, Lance Fortnow, Ashish Naik, and John D. Rogers. On inverting onto functions. Technical Report 95-05, University of Chicago, 1995.

[GG86] J. Geske and J. Grollmann. Relativizations of unambiguous and random polynomial time classes. *SIAM Journal on Computing*, 15(2):511–519, 1986.

[GS88] J. Grollmann and A Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–355, 1988.

[HH87] J. Hartmanis and L. Hemachandra. One-way functions, robustness and the nonisomorphism of NP-complete sets. In *Proceedings of the 2nd IEEE Structure in Complexity Theory Conference*, pages 160–173. IEEE, New York, 1987.

[HS92] S. Homer and A. Selman. Oracles for structural properties: The isomorphism problem and public-key cryptography. *Journal of Computer and System Sciences*, 44(2):287–301, 1992.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, Mass., 1979.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd IEEE Structure in Complexity Theory Conference*, pages 29–38. IEEE, New York, 1988.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 44–61. ACM, New York, 1989.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 157–166. ACM, New York, 1989.

[Meh73] Kurt Mehlhorn. On the size of computable functions. In *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pages 190–196. IEEE, New York, 1973.

[MV94] Andrei A. Muchnik and Nikolai K. Vereshchagin. A general method to construct oracles realizing given relationships between complexity classes. Technical Report 500, University of Rochester, 1994.

[Rac82] C. Rackoff. Relativized questions involving probablistic algorithms. *Journal of the ACM*, 29(1):261–268, 1982.

[Rog95] John D. Rogers. *Isomorphisms, Separability, and One-way Functions*. PhD thesis, University of Chicago, 1995.

[Ver93] N. Vereshchagin. Relationships between NP-sets, Co-NP-sets and P-sets relative to random oracles. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 132–138. IEEE, New York, 1993.